WEEK 4

How To Access Databases Using Python

Module Objectives:

- Explain how to use Python to connect to databases
- Create tables, load data, and query data using SQL
- Analyze and Visualize Data in Jupyter Notebooks

Lab Assignments:

- Connect to a database instance in the Cloud
- Query the data using SQL
- Analyze the data using Python

In This Video...

- Explain the basic concepts for connecting Python apps to a database
- Describe SQL APIs
- List SQL APIs for popular RDBMSes

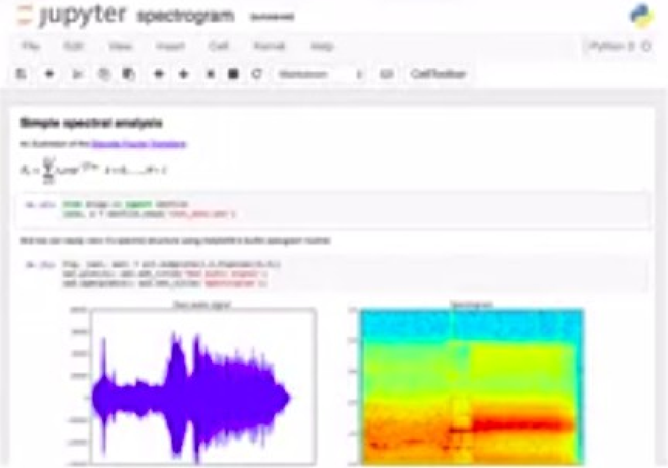
Benefits of python for database programming

- Python ecosystem : NumPy, pandas, matplotlib, SciPy
- Ease of use
- Portable
- Python supports relational database systems
- Python Database API (DB-API)
- Detailed documentation

Introduction to notebooks

Matlab notebook

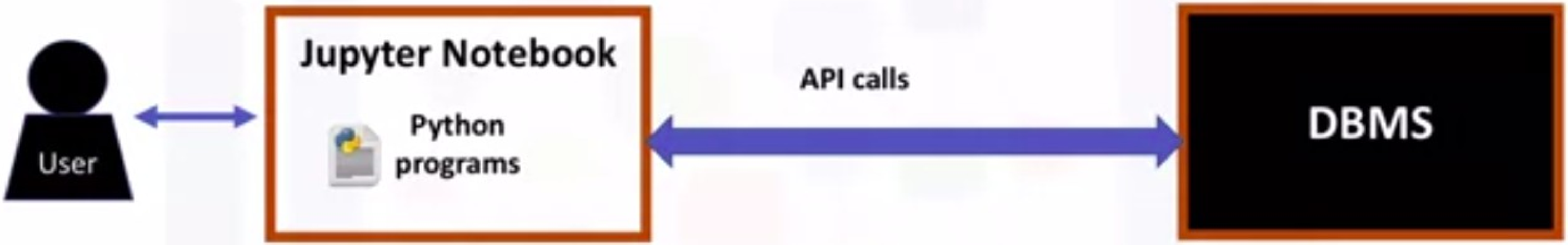
Jupyter notebook



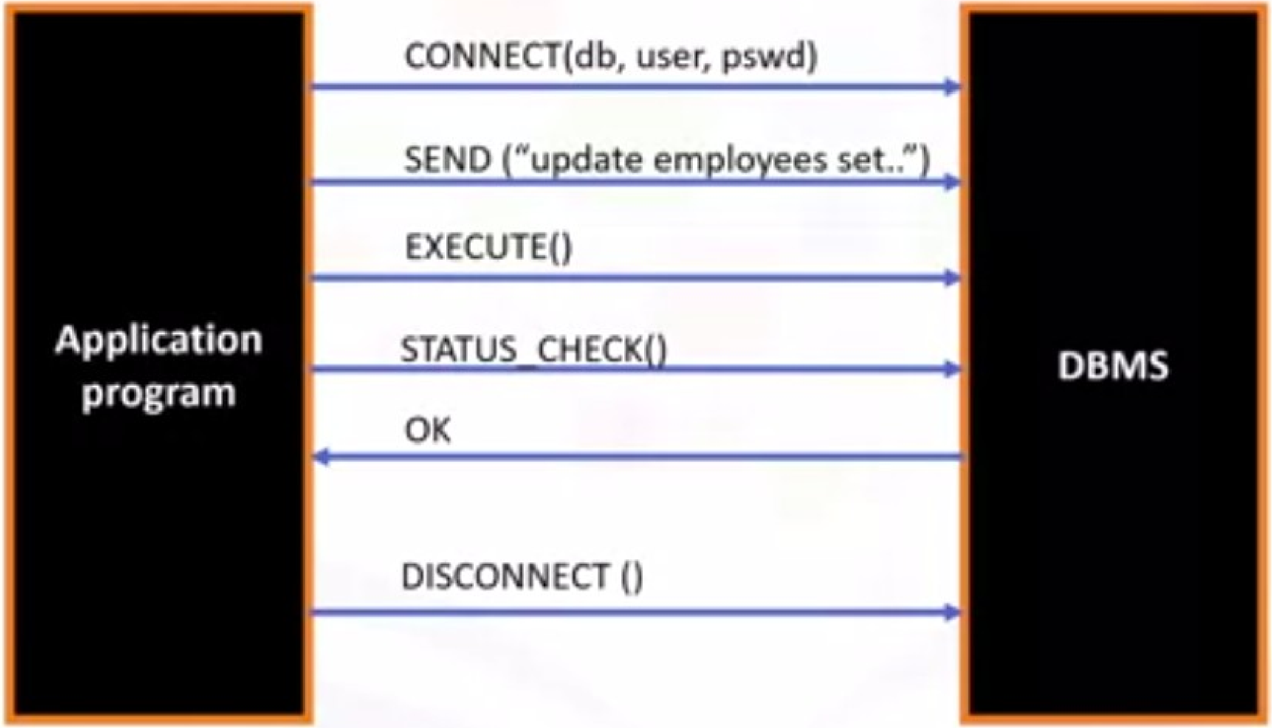
What are Jupyter Notebooks?

- Language of choice
- Share notebooks
- Interactive output
- Big data integration

Accessing databases using Python



What is a SQL API?

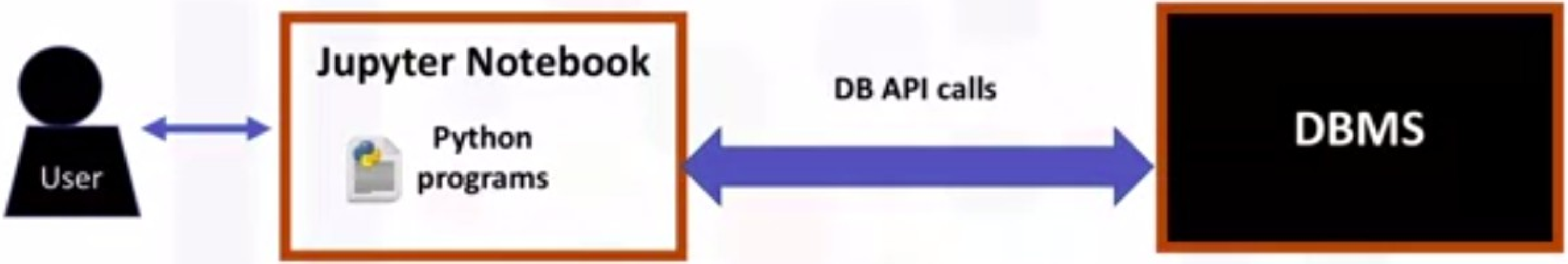


APIs used by popular SQL-based DBMS systems

Application or Database	SQL API
MySQL	MySQL C API
PostgreSQL	psycopg2
IBM DB2	ibm_db
SQL Server	dblib API
Database access for Microsoft Windows OS	ODBC
Oracle	OCI
Java	JDBC

Writing Code Using DB-API

What is a DB-API?



- Python's standard API for accessing relational databases
- Allows a single program that to work with multiple kinds of relational databases
- Learn DB-API functions once, use them with any database

Benefits of using DB-API

- Easy to implement and understand
- Encourages similarity between the Python modules used to access databases
- Achieves consistency
- Portable across databases
- Broad reach of database connectivity from Python

Examples of libraries used by database systems to connect to Python applications

Database	DB API
IBM Db2	Ibm_db
Compose for MySQL	MySQL Connector/Python
Compose for PostgreSQL	psycopg2
Compose for MongoDB	PyMongo

Concepts of the Python DB API

Connection Objects

- Database connections
- Manage transactions

Cursor Objects

- Database Queries
- Scroll through result set
- Retrieve results

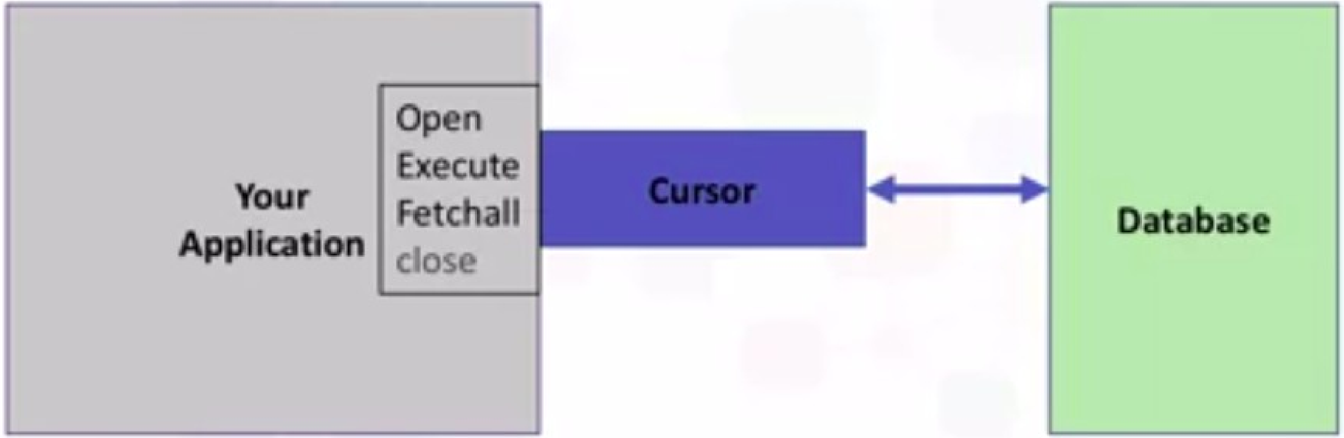
What are Connection methods?

- .cursor()
- .commit()
- .rollback()
- .close()

What are cursor methods?

- .callproc()
- .execute()
- .executemany()
- .fetchone()
- .fetchmany()
- .fetchall()
- .nextset()
- .arraysize()
- .close()

What is a database cursor?



Writing code using DB-API

```
from dbmodule import connect

#Create connection object

Connection =
connect(‘databasename’,
’username’, ’pswd’)


#Create a cursor object

Cursor=connection.cursor()
```

```
#Run Queries

Cursor.execute(‘select *
from mytable’)

Results=cursor.fetchall()


#Free resources

Cursor.close()

Connection.close()
```

Connecting to a database using ibm_db API

Overview

At the end of this lesson, you will be able to:

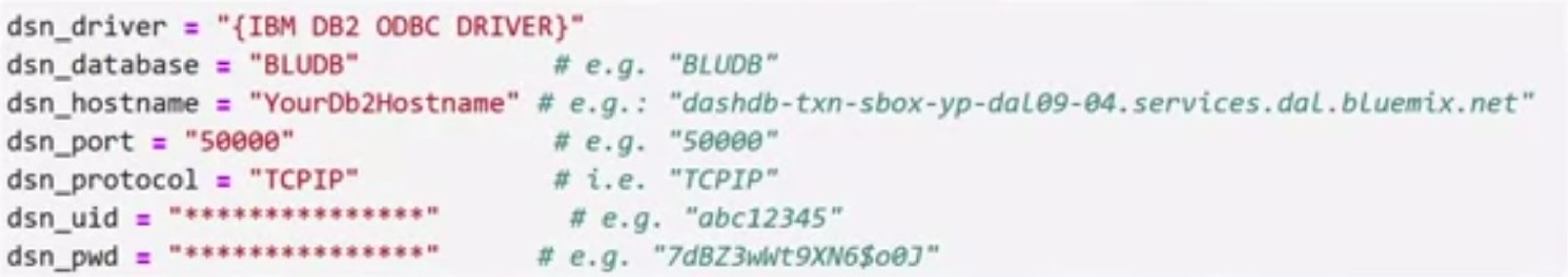
- Describe the ibm_db API
- List the credentials required to connect to a database
- Connect to an IBM db2 database using Python

What is ibm_db?

- The ibm_db API provides a variety of useful Python functions for accessing and manipulating data in an IBM data server Database

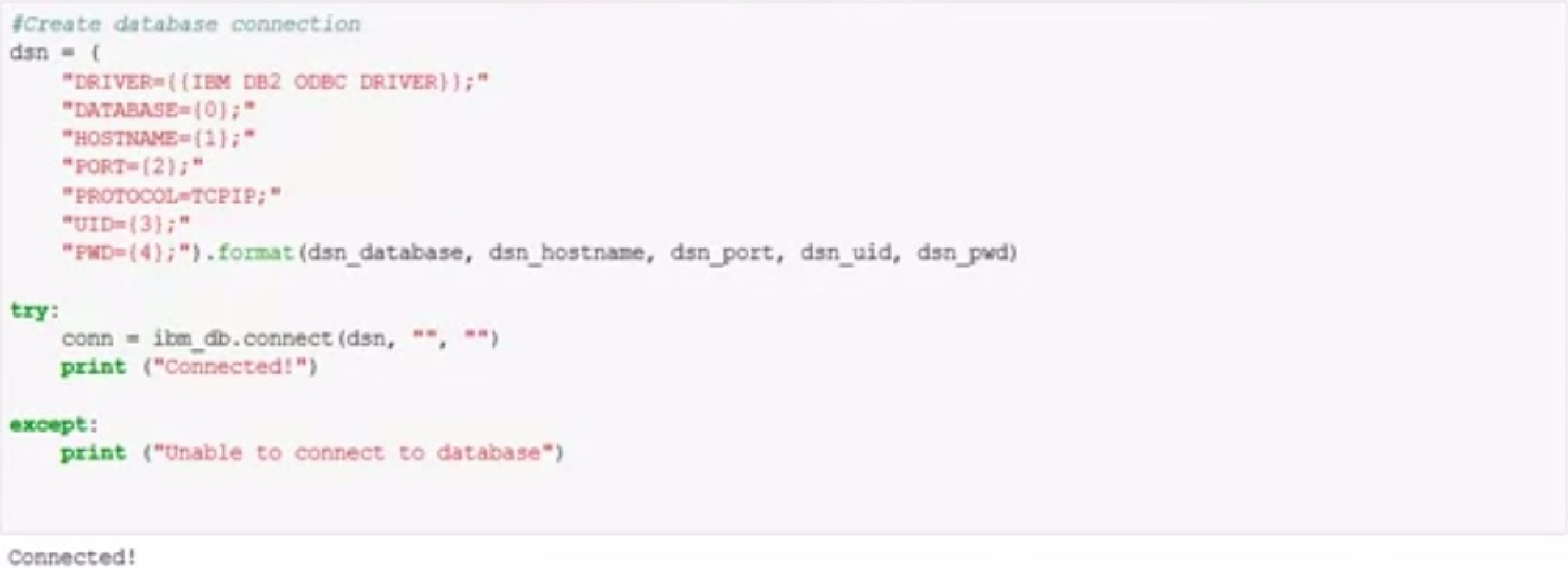
- Ibm_db API uses the IBM Data Server Driver for ODBC and CLI APIs to connect to IBM DB2 and Informix

Identify database connection credentials



```
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"            # e.g. "BLUDB"
dsn_hostname = "YourDb2Hostname" # e.g.: "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net"
dsn_port = "50000"                # e.g. "50000"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
dsn_uid = "***************"        # e.g. "abc12345"
dsn_pwd = "***************"      # e.g. "7dBZ3wWt9XN6$o0J"
```

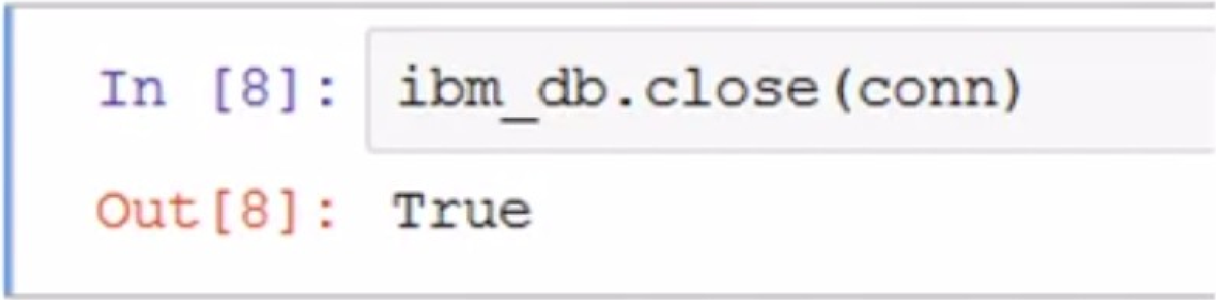
Create a database connection



```
#Create database connection
dsn = (
    "DRIVER={{IBM DB2 ODBC DRIVER}};"
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "PROTOCOL=TCPIP;"
    "UID={3};"
    "PWD={4};").format(dsn_database, dsn_hostname, dsn_port, dsn_uid, dsn_pwd)

try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected!")

except:
    print ("Unable to connect to database")
```

```
Connected!
```

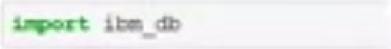
Close the database connection



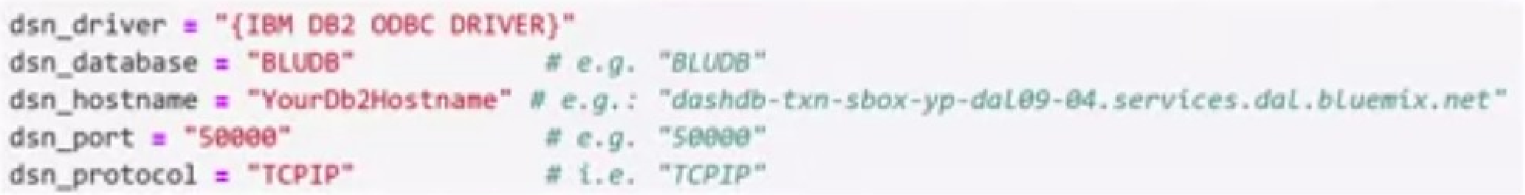
```
In [8]: ibm_db.close(conn)

Out[8]: True
```

Creating Tables, Loading Data and Querying Data

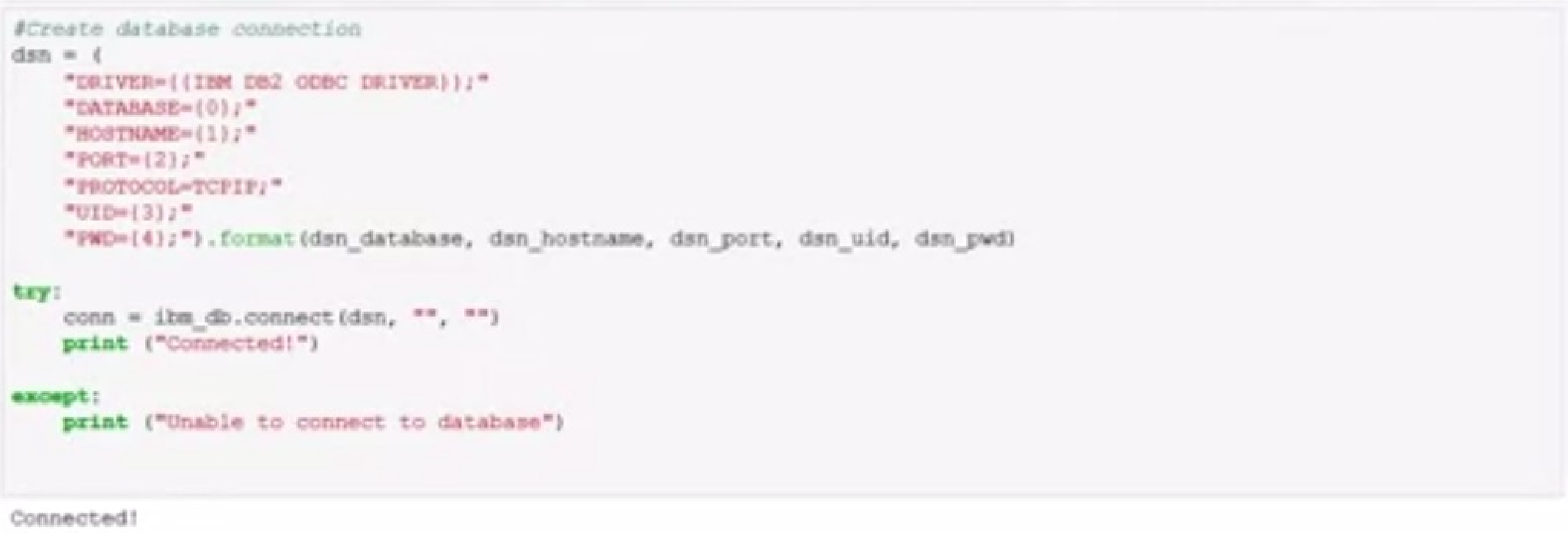
Connect to the database



```python
import ibm_db
```



```python
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"            # e.g. "BLUDB"
dsn_hostname = "YourDb2Hostname" # e.g.: "dashdb-txn-sbox-yp-dal09-04.services.dal.bluemix.net"
dsn_port = "50000"                # e.g. "50000"
dsn_protocol = "TCPIP"            # i.e. "TCPIP"
```



```python
#Create database connection
dsn = (
    "DRIVER={{IBM DB2 ODBC DRIVER}};"
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "PROTOCOL=TCPIP;"
    "UID={3};"
    "PWD={4};").format(dsn_database, dsn_hostname, dsn_port, dsn_uid, dsn_pwd)

try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected!")

except:
    print ("Unable to connect to database")
```

```
Connected!
```

Creating tables

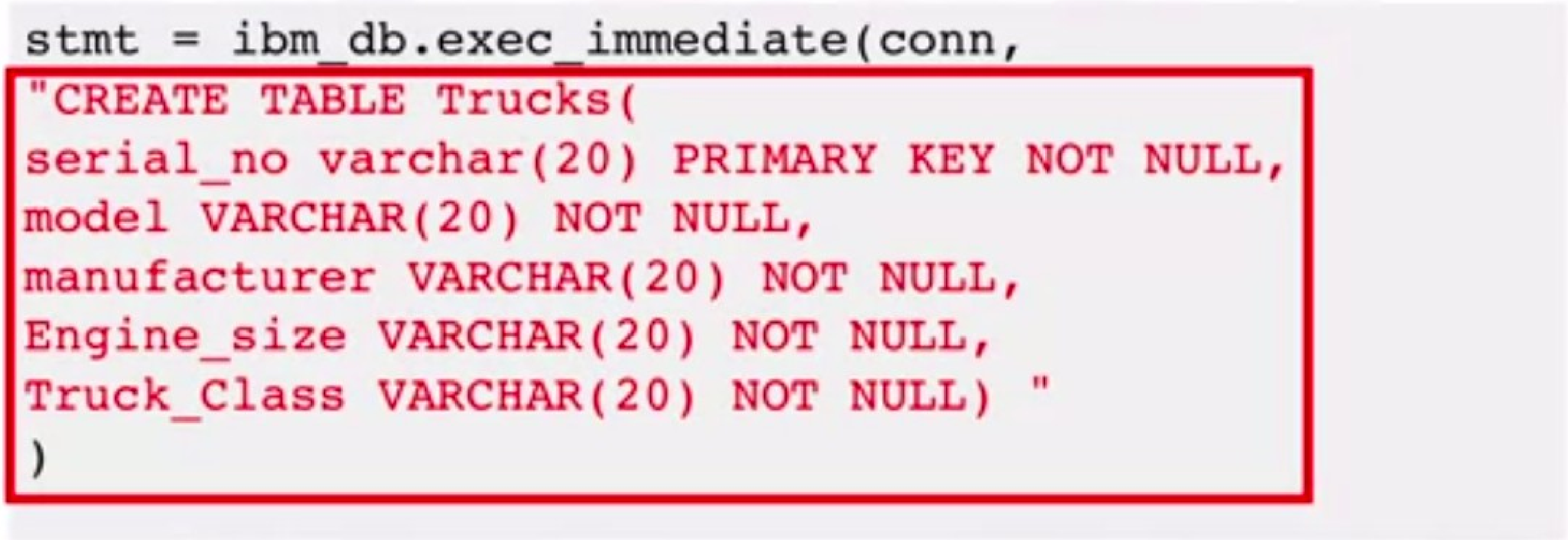
Serial No	Model	Manufacturer	Engine Size	Class
A1234	Lonestar	International Trucks	Cummins ISX15	Class 8
B5432	Volvo VN	Volvo Trucks	Volvo D11	Heavy Duty Tractor Class 8
C5674	Kenworth W900	Kenworth Truck Company	Caterpillar C9	Class 8

ibm_db.exec_immediate()

The parameters for the function are:

- Connection
- Statement
- Options

Python code to create a table

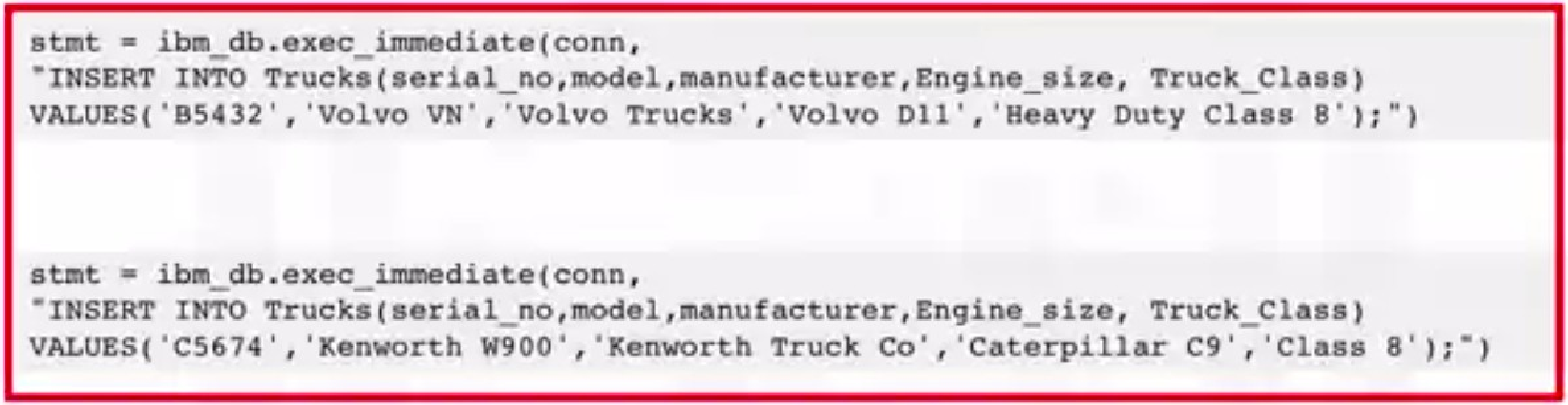


```
stmt = ibm_db.exec_immediate(conn,
"CREATE TABLE Trucks(
serial_no varchar(20) PRIMARY KEY NOT NULL,
model VARCHAR(20) NOT NULL,
manufacturer VARCHAR(20) NOT NULL,
Engine_size VARCHAR(20) NOT NULL,
Truck_Class VARCHAR(20) NOT NULL) "
)
```

Python code to insert data into the table

```
stmt = ibm_db.exec_immediate(conn,
"INSERT INTO Trucks(serial_no,
model,manufacturer,Engine_size, Truck_Class)
VALUES('A1234','Lonestar','International
Trucks’,'Cummins ISX15','Class 8’);")
```

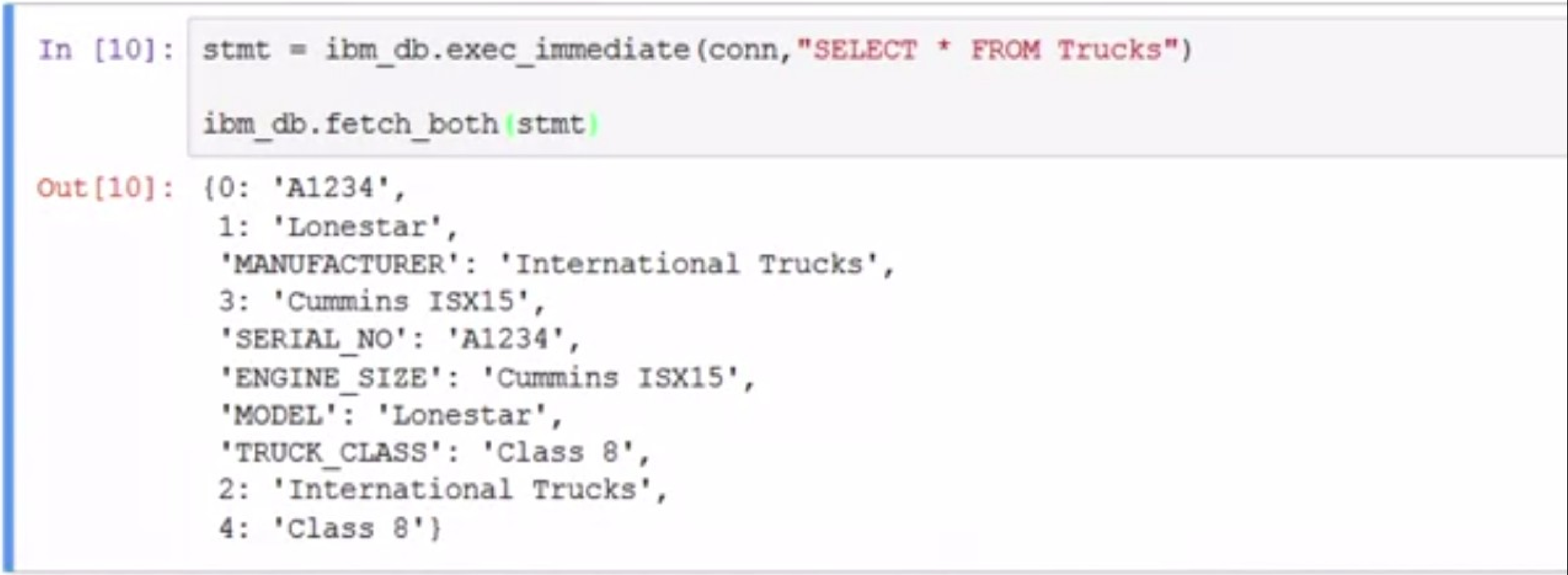
Insert more rows to the table



```
stmt = ibm_db.exec_immediate(conn,
"INSERT INTO Trucks(serial_no,model,manufacturer,Engine_size, Truck_Class)
VALUES('B5432','Volvo VN','Volvo Trucks','Volvo D11','Heavy Duty Class 8');")
```

```
stmt = ibm_db.exec_immediate(conn,
"INSERT INTO Trucks(serial_no,model,manufacturer,Engine_size, Truck_Class)
VALUES('C5674','Kenworth W900','Kenworth Truck Co','Caterpillar C9','Class 8');")
```

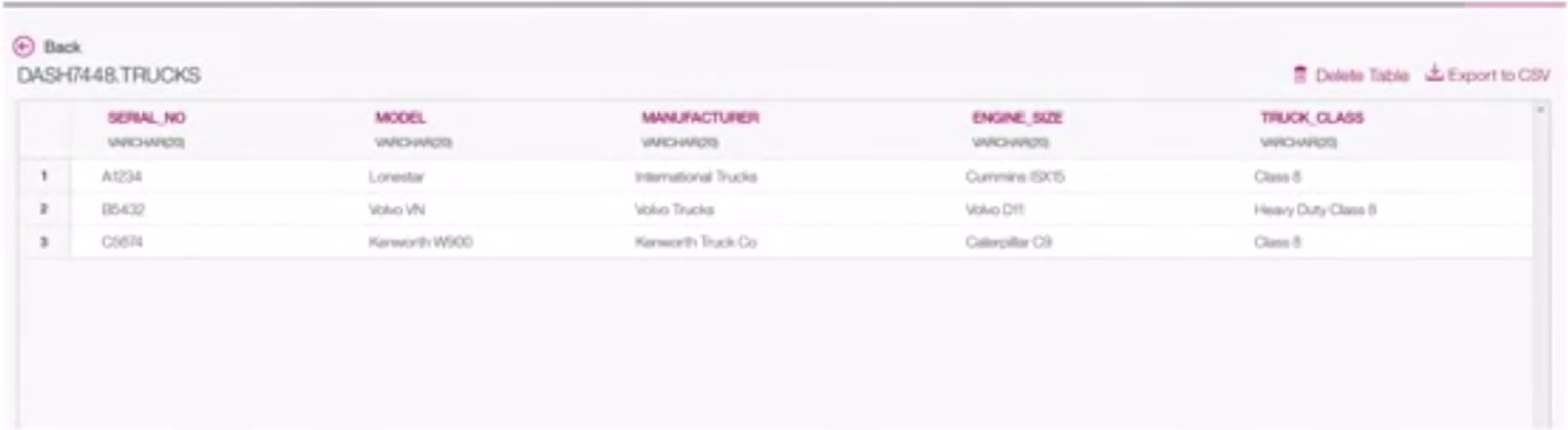
Python code to query data



```
In [10]:  stmt = ibm_db.exec_immediate(conn,"SELECT * FROM Trucks")

          ibm_db.fetch_both(stmt)
```

```
Out[10]:  {0: 'A1234',
           1: 'Lonestar',
           'MANUFACTURER': 'International Trucks',
           3: 'Cummins ISX15',
           'SERIAL_NO': 'A1234',
           'ENGINE_SIZE': 'Cummins ISX15',
           'MODEL': 'Lonestar',
           'TRUCK_CLASS': 'Class 8',
           2: 'International Trucks',
           4: 'Class 8'}
```

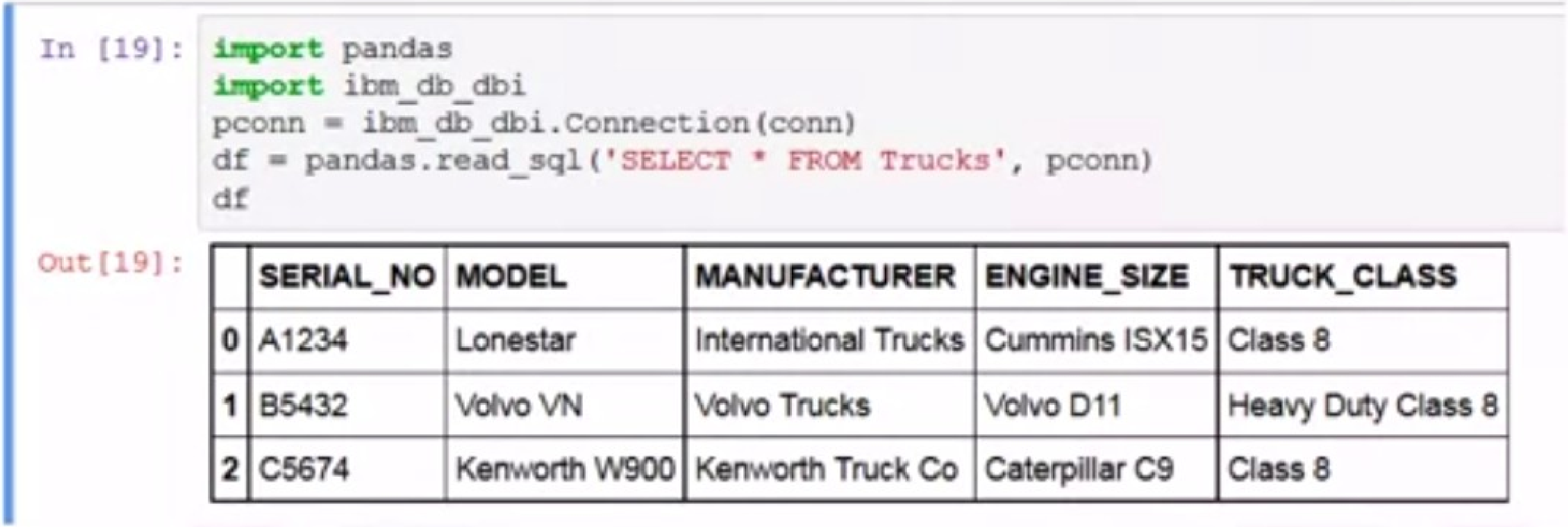
Confirm the output on Db2 on Cloud console



DASH7448.TRUCKS

	SERIAL_NO VARCHAR(20)	MODEL VARCHAR(20)	MANUFACTURER VARCHAR(20)	ENGINE_SIZE VARCHAR(20)	TRUCK_CLASS VARCHAR(20)
1	A1234	Lonestar	International Trucks	Cummins ISX15	Class 8
2	B5432	Volvo VN	Volvo Trucks	Volvo D11	Heavy Duty Class 8
3	C5674	Kenworth W900	Kenworth Truck Co	Caterpillar C9	Class 8

Using pandas



```
In [19]: import pandas
         import ibm_db_dbi
         pconn = ibm_db_dbi.Connection(conn)
         df = pandas.read_sql('SELECT * FROM Trucks', pconn)
         df
```

Out[19]:

	SERIAL_NO	MODEL	MANUFACTURER	ENGINE_SIZE	TRUCK_CLASS
0	A1234	Lonestar	International Trucks	Cummins ISX15	Class 8
1	B5432	Volvo VN	Volvo Trucks	Volvo D11	Heavy Duty Class 8
2	C5674	Kenworth W900	Kenworth Truck Co	Caterpillar C9	Class 8

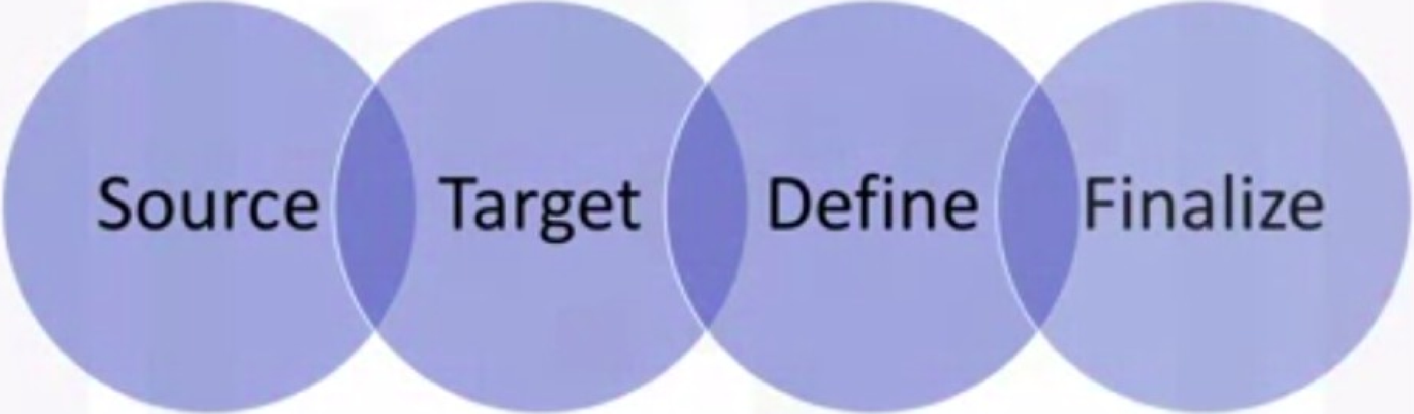
Analyzing data with Python

IBM Developer

SKILLS NETWORK

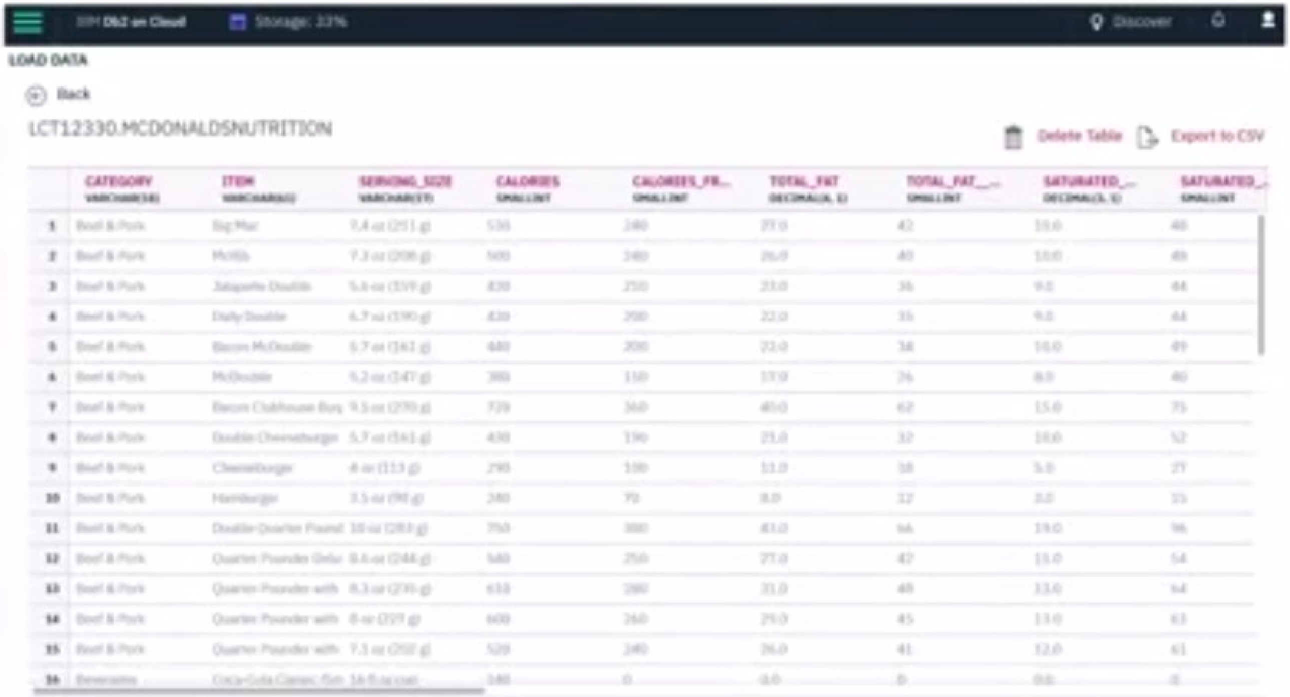
McDonald’s menu nutrition facts

Load csv file into DB2 on cloud



Load csv file into DB2 on cloud

- Source
- Target
- Define
- Finalize



LOAD DATA

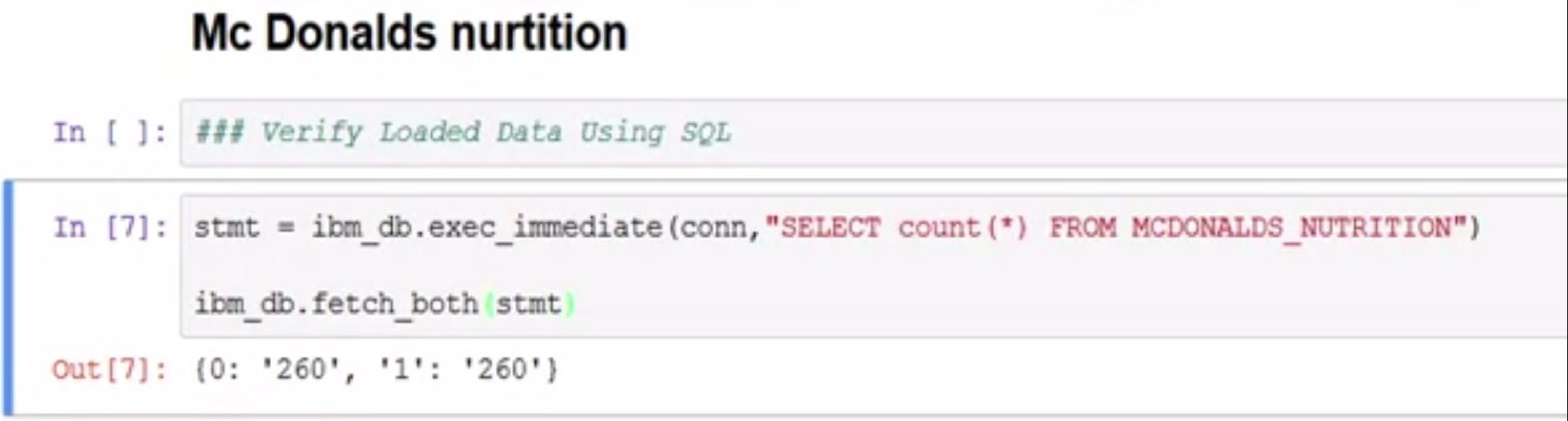
Back

LCT12330.MCDONALDSNUTRITION

Delete Table | Export to CSV

	CATEGORY VARCHAR(18)	ITEM VARCHAR(62)	SERVING_SIZE VARCHAR(27)	CALORIES SMALLINT	CALORIES_FR... SMALLINT	TOTAL_FAT DECIMAL(4, 1)	TOTAL_FAT__... SMALLINT	SATURATED_... DECIMAL(3, 1)	SATURATED_... SMALLINT
1	Beef & Pork	Big Mac	7.4 oz (211 g)	530	240	27.0	42	10.0	48
2	Beef & Pork	McRib	7.3 oz (208 g)	500	240	26.0	40	10.0	48
3	Beef & Pork	Jalapeño Double	5.6 oz (159 g)	430	210	23.0	36	9.0	44
4	Beef & Pork	Daily Double	6.7 oz (190 g)	430	200	22.0	35	9.0	44
5	Beef & Pork	Bacon McDouble	5.7 oz (161 g)	440	200	22.0	34	10.0	49
6	Beef & Pork	McDouble	5.2 oz (147 g)	380	150	17.0	26	8.0	40
7	Beef & Pork	Bacon Clubhouse Burg	9.5 oz (270 g)	720	360	40.0	62	15.0	75
8	Beef & Pork	Double Cheeseburger	5.7 oz (161 g)	430	190	21.0	32	10.0	52
9	Beef & Pork	Cheeseburger	4 oz (113 g)	290	100	11.0	18	5.0	27
10	Beef & Pork	Hamburger	3.5 oz (98 g)	240	70	8.0	12	3.0	15
11	Beef & Pork	Double Quarter Pound	10 oz (283 g)	750	380	43.0	66	19.0	96
12	Beef & Pork	Quarter Pounder Delu	8.6 oz (244 g)	540	250	27.0	42	11.0	54
13	Beef & Pork	Quarter Pounder with	8.3 oz (235 g)	610	280	31.0	48	13.0	64
14	Beef & Pork	Quarter Pounder with	8 oz (227 g)	600	260	29.0	45	13.0	63
15	Beef & Pork	Quarter Pounder with	7.1 oz (202 g)	520	240	26.0	41	12.0	61
16	Beverages	Coca-Cola Classic (Sm	16 fl oz cup	140	0	0.0	0	0.0	0

Verify Loaded Data Using SQL

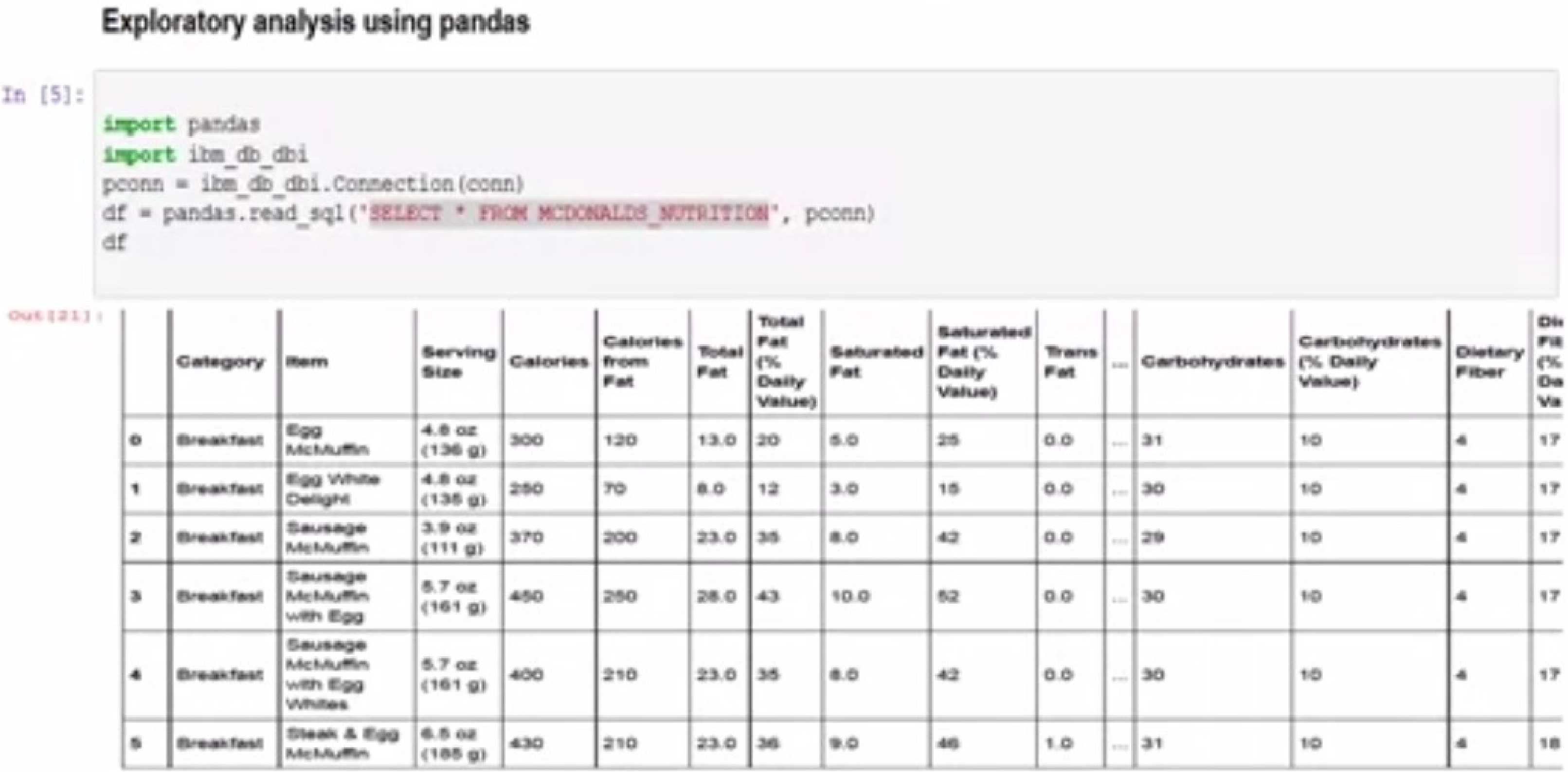


Mc Donalds nurtition

```
In [ ]: ### Verify Loaded Data Using SQL
```

```
In [7]: stmt = ibm_db.exec_immediate(conn,"SELECT count(*) FROM MCDONALDS_NUTRITION")

        ibm_db.fetch_both(stmt)

Out[7]: {0: '260', '1': '260'}
```

Using pandas



Exploratory analysis using pandas

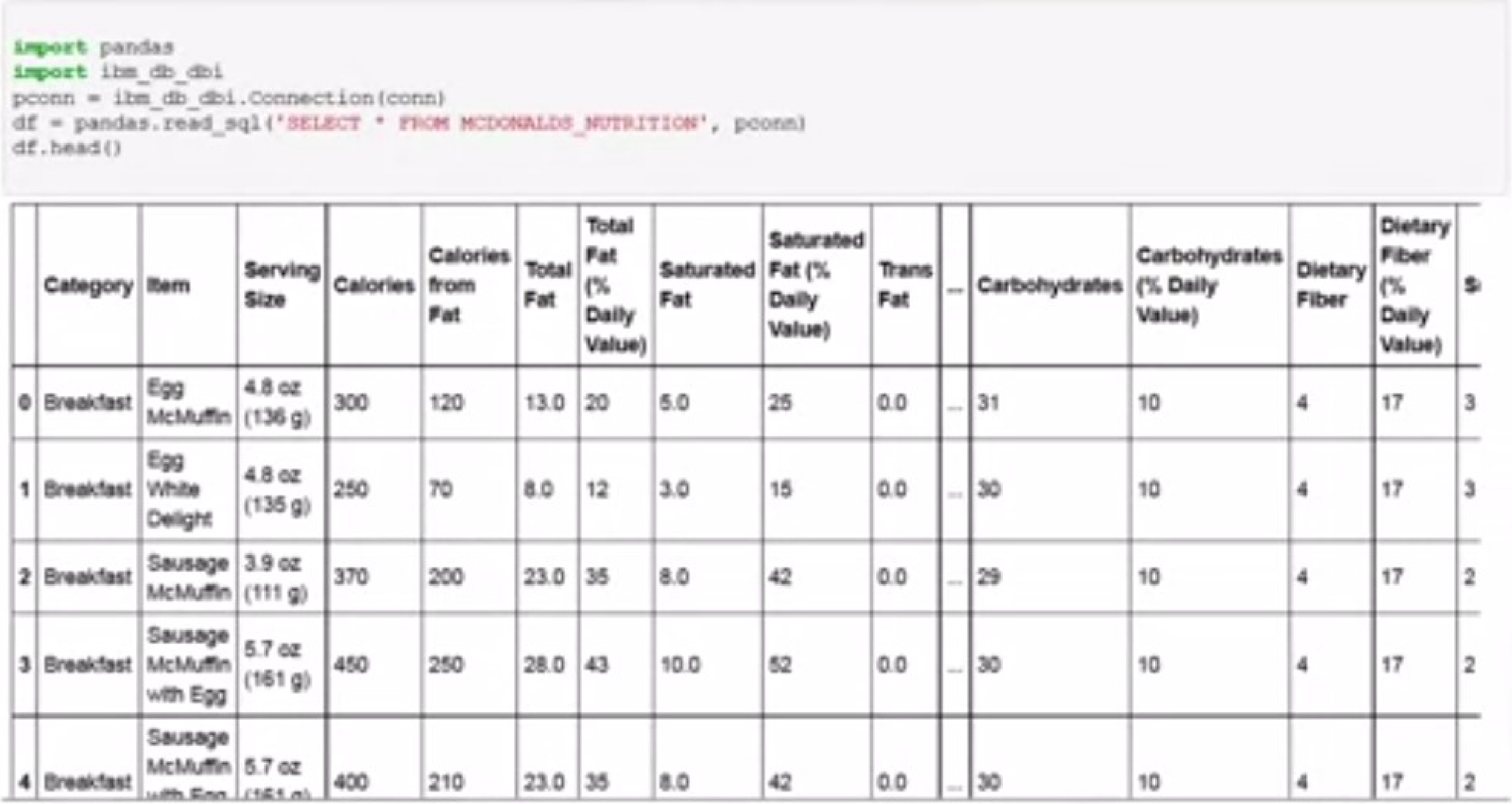
In [5]:

```
import pandas
import ibm_db_dbi
pconn = ibm_db_dbi.Connection(conn)
df = pandas.read_sql('SELECT * FROM MCDONALDS_NUTRITION', pconn)
df
```

Out[21]:

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carbohydrates	Carbohydrates (% Daily Value)	Dietary Fiber	Di Fi (% Da Va
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	20	5.0	25	0.0	...	31	10	4	17
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	12	3.0	15	0.0	...	30	10	4	17
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	35	8.0	42	0.0	...	29	10	4	17
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	43	10.0	52	0.0	...	30	10	4	17
4	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210	23.0	35	8.0	42	0.0	...	30	10	4	17
5	Breakfast	Steak & Egg McMuffin	6.5 oz (185 g)	430	210	23.0	36	9.0	46	1.0	...	31	10	4	18

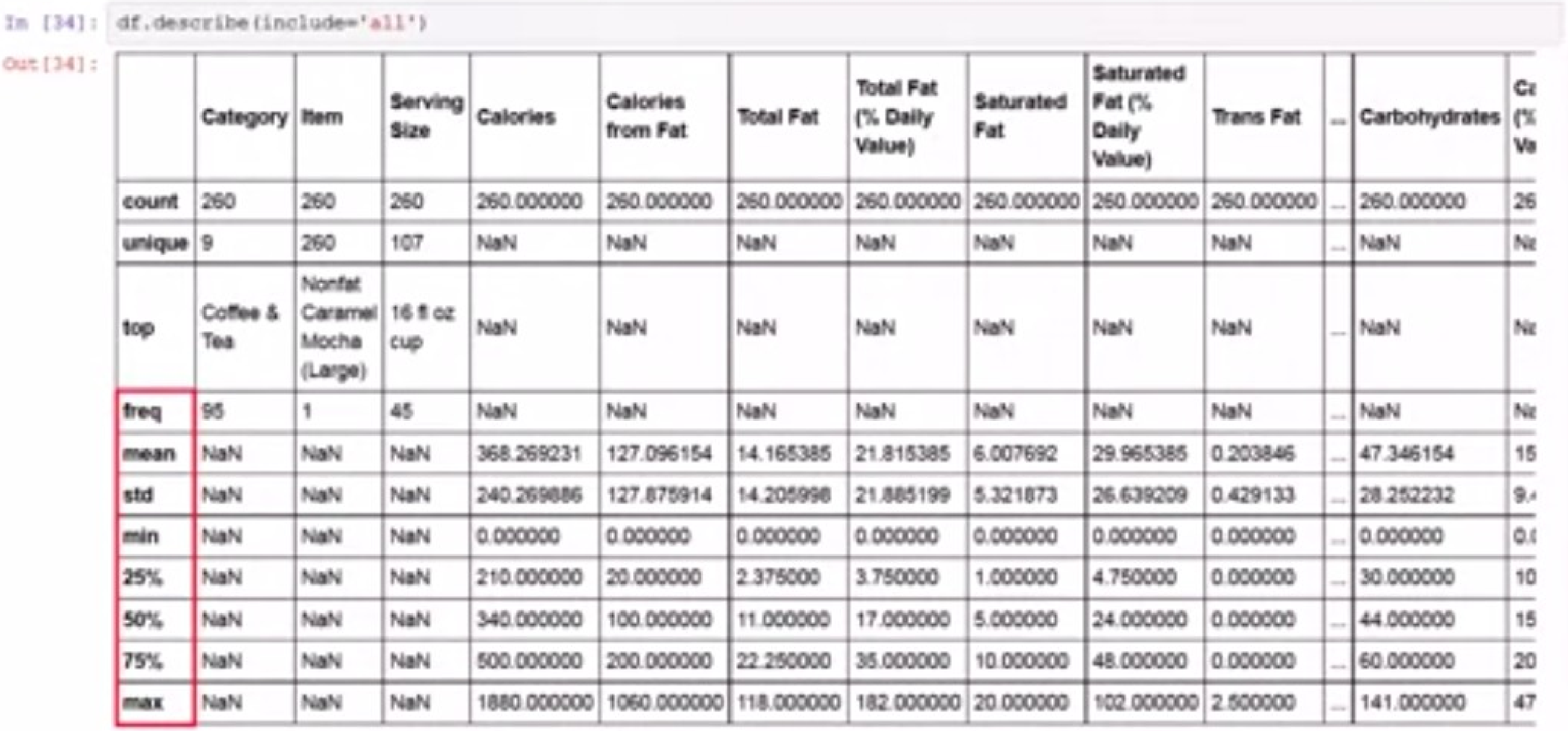
View first few rows



```
import pandas
import ibm_db_dbi
pconn = ibm_db_dbi.Connection(conn)
df = pandas.read_sql('SELECT * FROM MCDONALDS_NUTRITION', pconn)
df.head()
```

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carbohydrates	Carbohydrates (% Daily Value)	Dietary Fiber	Dietary Fiber (% Daily Value)	S
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	20	5.0	25	0.0	...	31	10	4	17	3
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	12	3.0	15	0.0	...	30	10	4	17	3
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	35	8.0	42	0.0	...	29	10	4	17	2
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	43	10.0	52	0.0	...	30	10	4	17	2
4	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	400	210	23.0	35	8.0	42	0.0	...	30	10	4	17	2

Learn about your data



```
In [34]: df.describe(include='all')
```

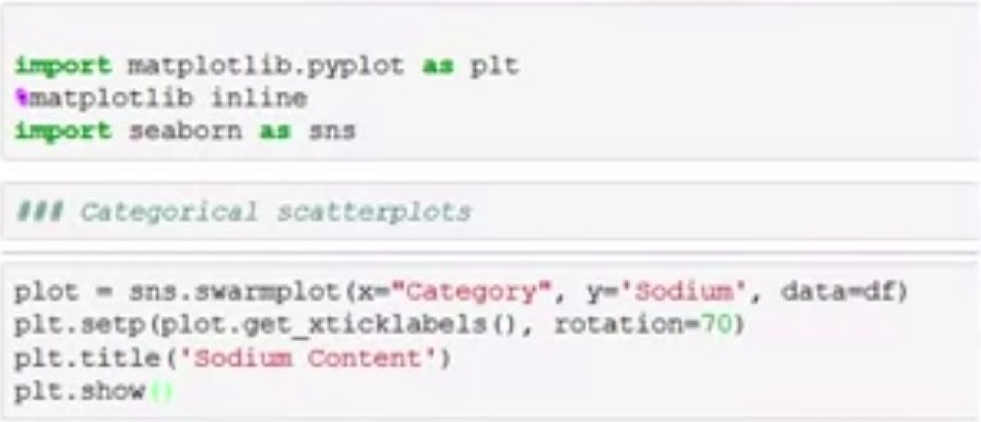
Out[34]:

	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carbohydrates	Ca (% Va
count	260	260	260	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000	260.000000	...	260.000000	26
unique	9	260	107	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
top	Coffee & Tea	Nonfat Caramel Mocha (Large)	16 fl oz cup	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
freq	95	1	45	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	Na
mean	NaN	NaN	NaN	368.269231	127.096154	14.165385	21.815385	6.007692	29.965385	0.203846	...	47.346154	15
std	NaN	NaN	NaN	240.269886	127.875914	14.205998	21.885199	5.321873	26.639209	0.429133	...	28.252232	9.4
min	NaN	NaN	NaN	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0
25%	NaN	NaN	NaN	210.000000	20.000000	2.375000	3.750000	1.000000	4.750000	0.000000	...	30.000000	10
50%	NaN	NaN	NaN	340.000000	100.000000	11.000000	17.000000	5.000000	24.000000	0.000000	...	44.000000	15
75%	NaN	NaN	NaN	500.000000	200.000000	22.250000	35.000000	10.000000	48.000000	0.000000	...	60.000000	20
max	NaN	NaN	NaN	1880.000000	1060.000000	118.000000	182.000000	20.000000	102.000000	2.500000	...	141.000000	47

Which food item has maximum sodium content?

- Main source of sodium is table salt
- Average American eats 5 teaspoons/day
- Sodium mostly added during preparation
- Foods that don't taste salty may be high in sodium
- Sodium controls fluid balance in our bodies
- Too much sodium may raise blood pressure
- Target less than 2,000 milligrams/day

Which food item has maximum sodium content?

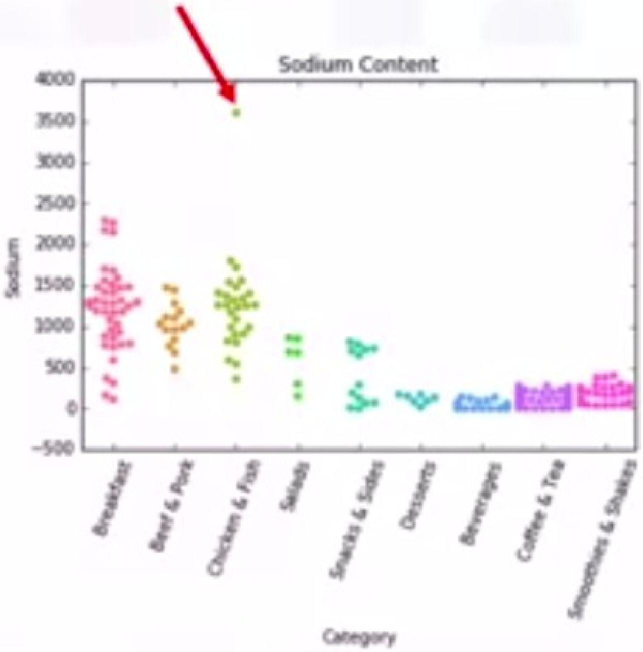


```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
### Categorical scatterplots
```

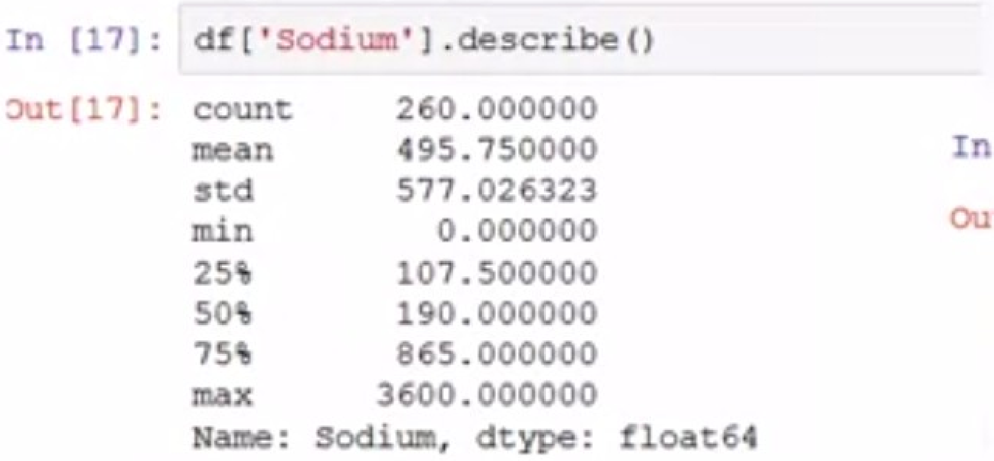
```
plot = sns.swarmplot(x="Category", y='Sodium', data=df)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title('Sodium Content')
plt.show()
```

Notice this high sodium value



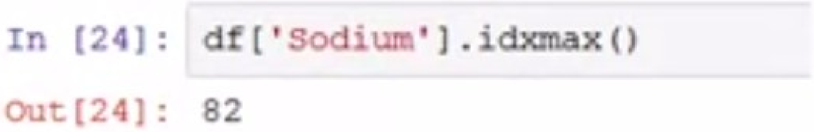
Which food item has maximum sodium content?

Code 1



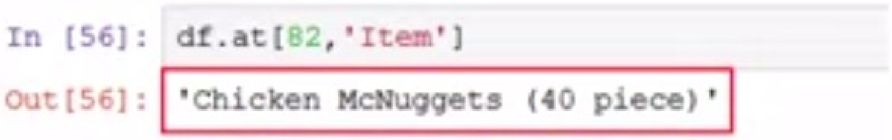
```
In [17]: df['Sodium'].describe()

Out[17]: count      260.000000
         mean       495.750000
         std        577.026323
         min          0.000000
         25%        107.500000
         50%        190.000000
         75%        865.000000
         max       3600.000000
         Name: Sodium, dtype: float64
```

Code 2



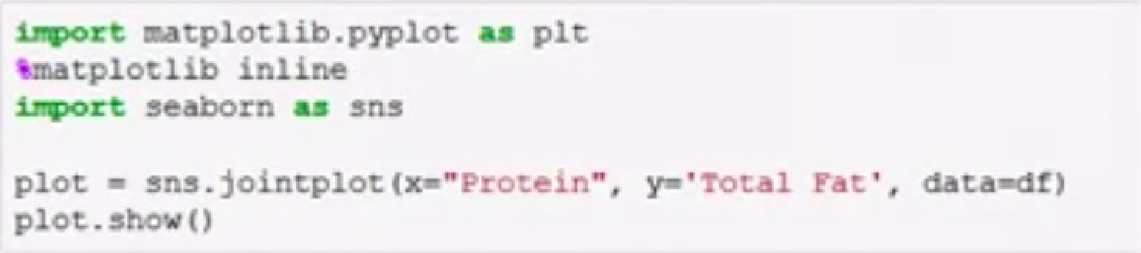
```
In [24]: df['Sodium'].idxmax()

Out[24]: 82
```

Code 3

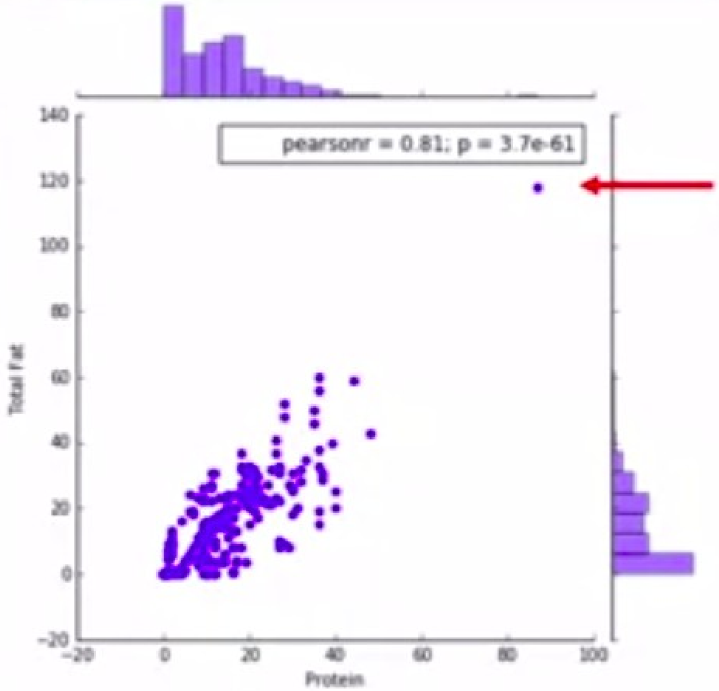


```
In [56]: df.at[82,'Item']

Out[56]: 'Chicken McNuggets (40 piece)'
```

Further data exploration using visualizations

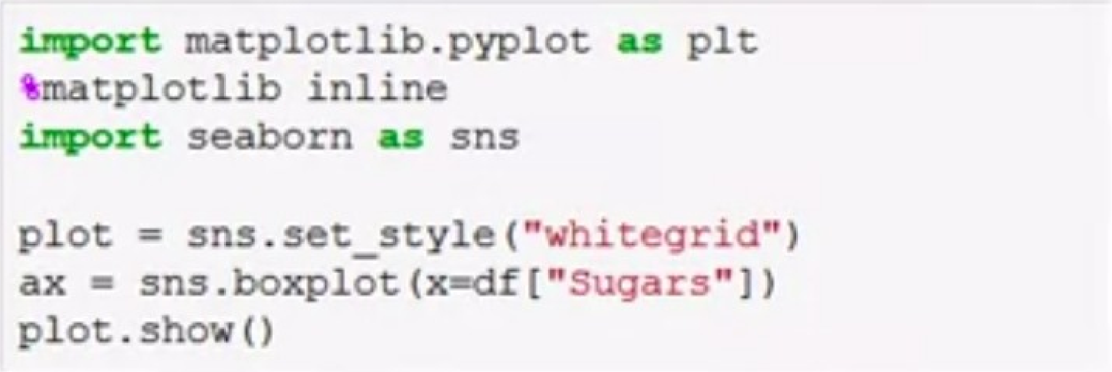


```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

plot = sns.jointplot(x="Protein", y='Total Fat', data=df)
plot.show()
```

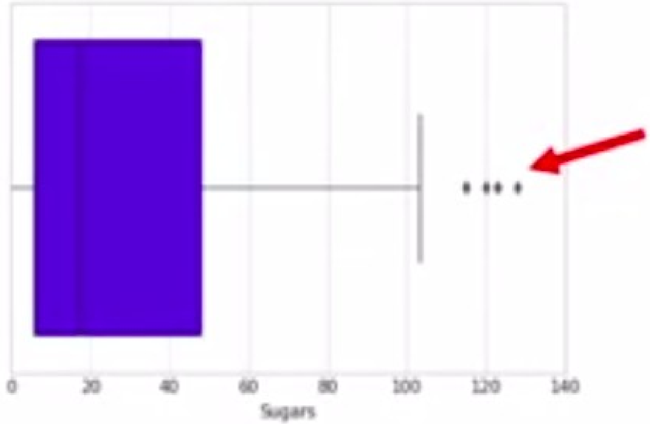


Outlier

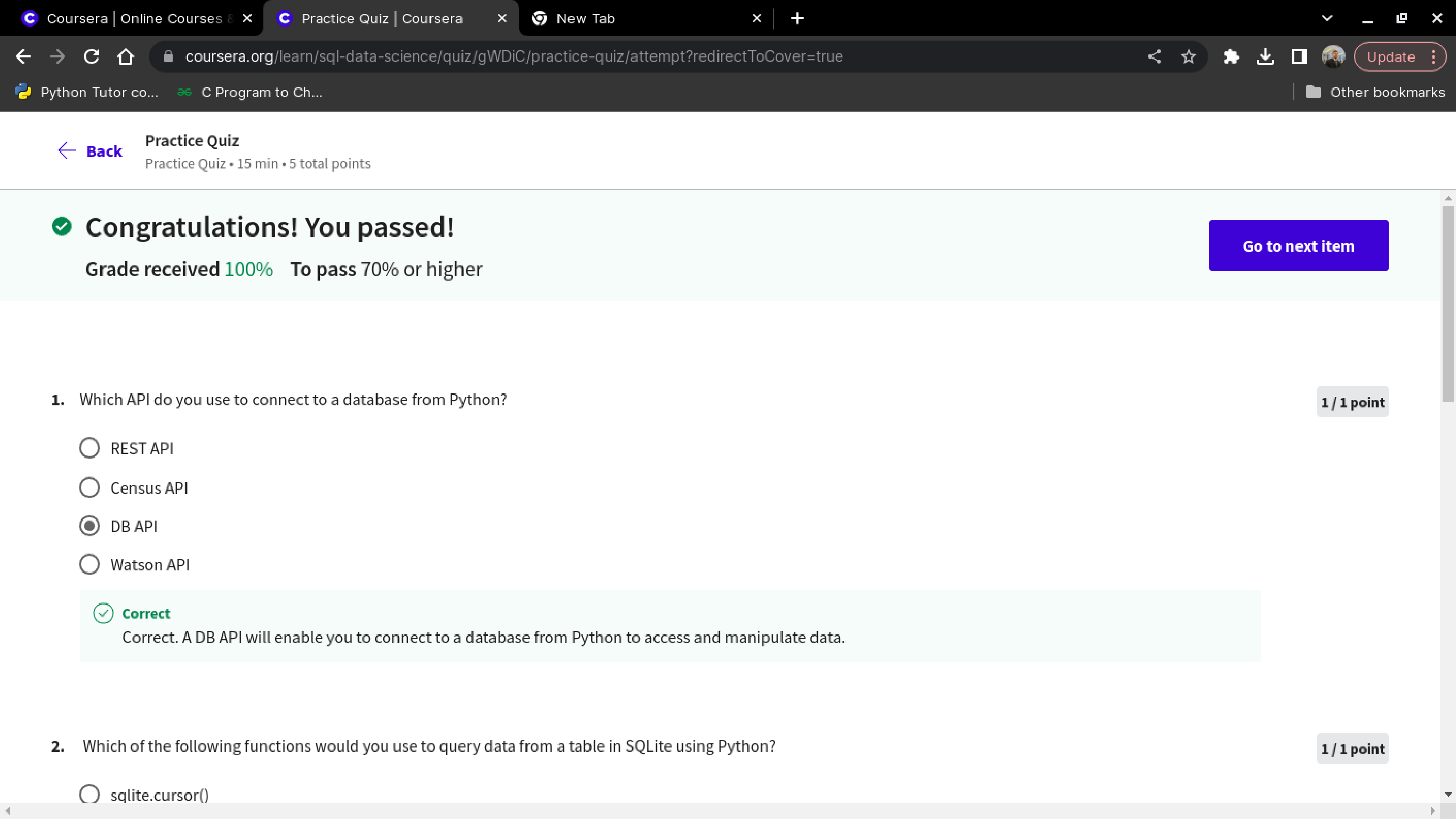
Further data exploration using visualizations



```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

plot = sns.set_style("whitegrid")
ax = sns.boxplot(x=df["Sugars"])
plot.show()
```



Outliers


Coursera | Online Courses &
Practice Quiz | Coursera
New Tab
coursera.org/learn/sql-data-science/quiz/gWDiC/practice-quiz/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks

← Back

Practice Quiz

Practice Quiz • 15 min • 5 total points

Congratulations! You passed!

Grade received 100% **To pass** 70% or higher

Go to next item

1. Which API do you use to connect to a database from Python?

1 / 1 point

○ REST API

○ Census API

● DB API

○ Watson API

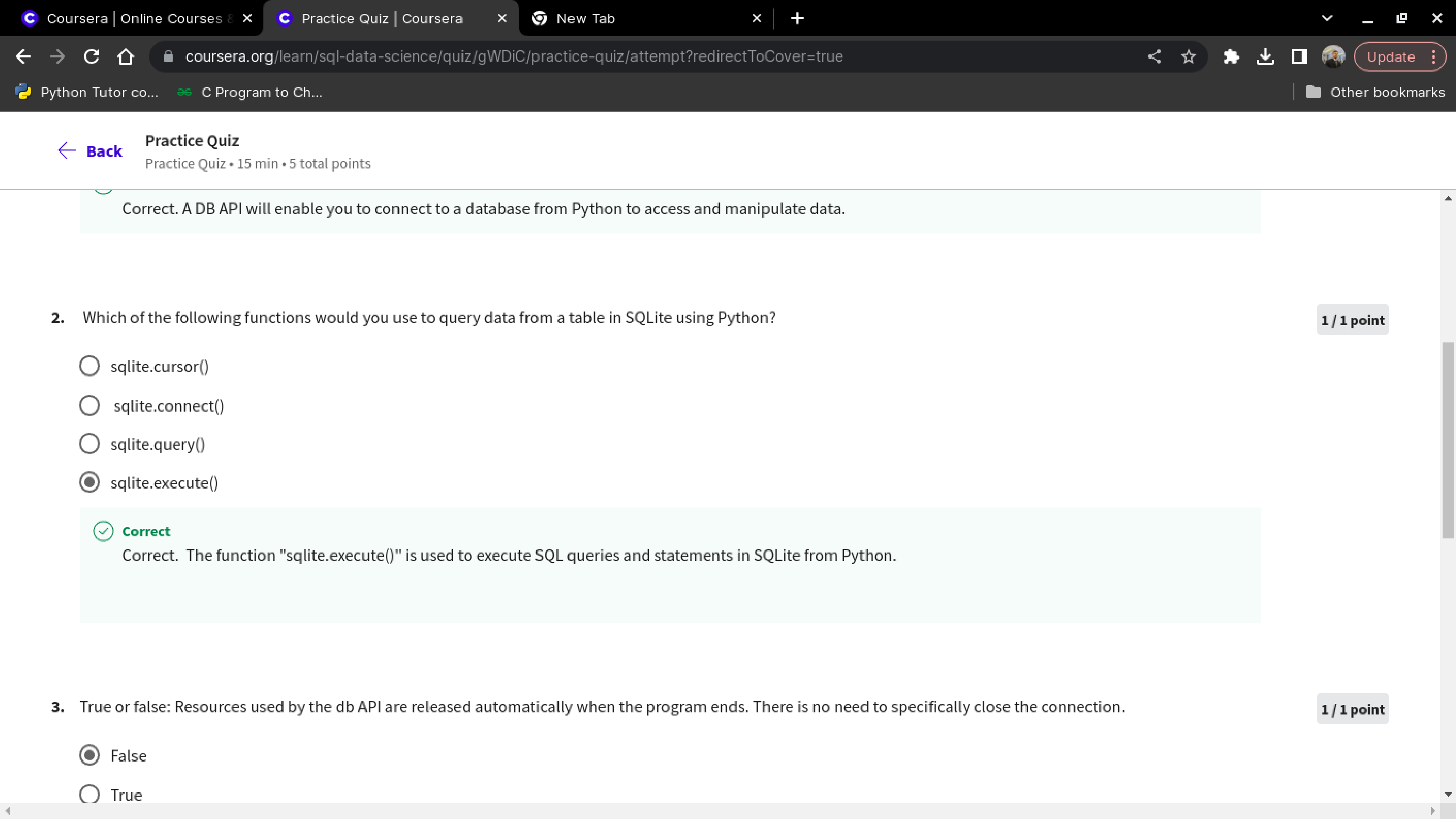
Correct

Correct. A DB API will enable you to connect to a database from Python to access and manipulate data.

2. Which of the following functions would you use to query data from a table in SQLite using Python?

1 / 1 point

○ sqlite.cursor()


Coursera | Online Courses &
Practice Quiz | Coursera
New Tab
coursera.org/learn/sql-data-science/quiz/gWDiC/practice-quiz/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks
Back

Practice Quiz

Practice Quiz • 15 min • 5 total points

Correct. A DB API will enable you to connect to a database from Python to access and manipulate data.

2. Which of the following functions would you use to query data from a table in SQLite using Python?

1 / 1 point

○ sqlite.cursor()

○ sqlite.connect()

○ sqlite.query()

● sqlite.execute()

Correct

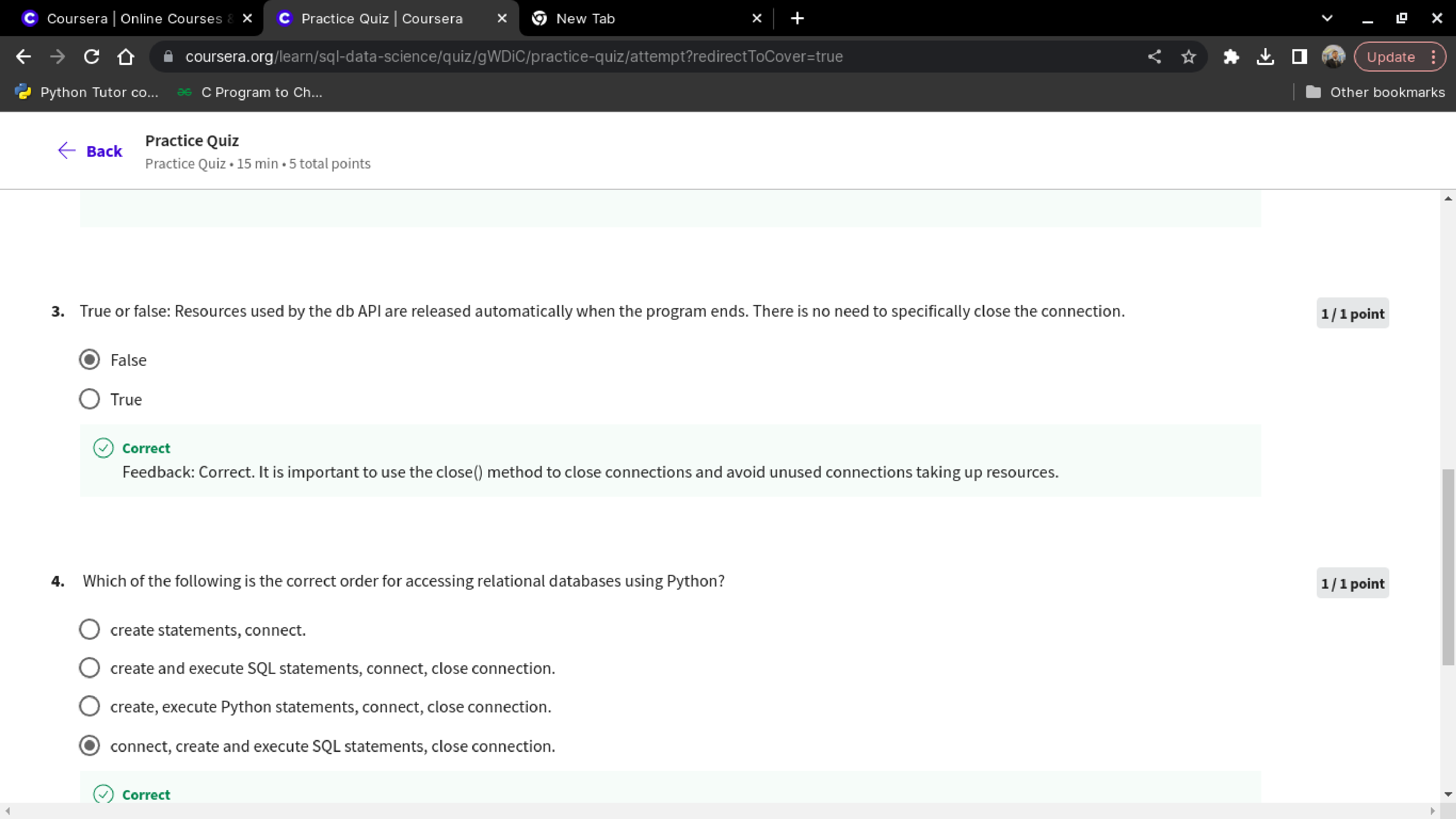
Correct. The function "sqlite.execute()" is used to execute SQL queries and statements in SQLite from Python.

3. True or false: Resources used by the db API are released automatically when the program ends. There is no need to specifically close the connection.

1 / 1 point

● False

○ True


Coursera | Online Courses &
Practice Quiz | Coursera
New Tab
coursera.org/learn/sql-data-science/quiz/gWDiC/practice-quiz/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks
Back

Practice Quiz

Practice Quiz • 15 min • 5 total points

3. True or false: Resources used by the db API are released automatically when the program ends. There is no need to specifically close the connection.

1 / 1 point

● False

○ True

Correct

Feedback: Correct. It is important to use the close() method to close connections and avoid unused connections taking up resources.

4. Which of the following is the correct order for accessing relational databases using Python?

1 / 1 point

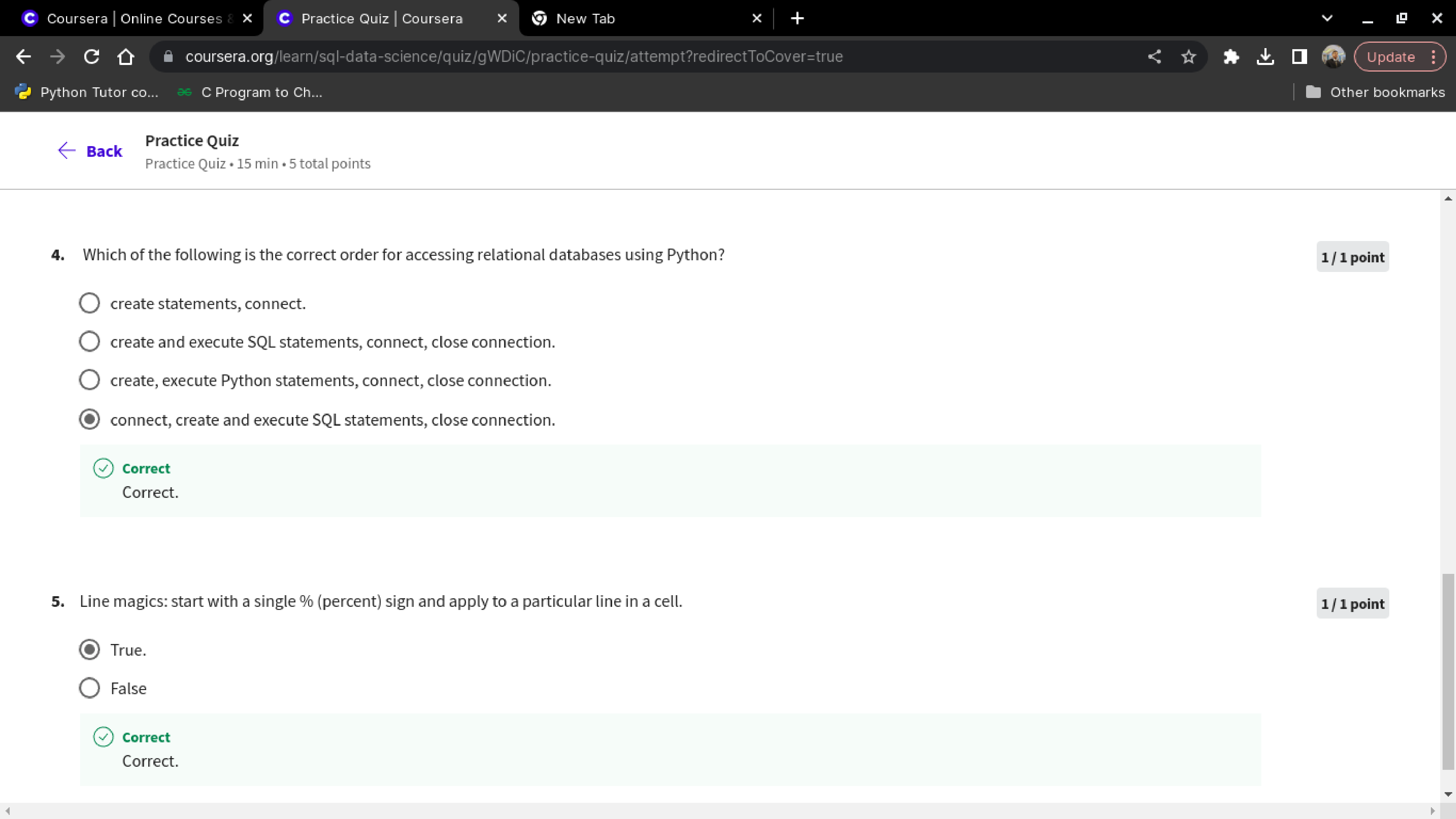
○ create statements, connect.

○ create and execute SQL statements, connect, close connection.

○ create, execute Python statements, connect, close connection.

● connect, create and execute SQL statements, close connection.

Correct


Coursera | Online Courses &
Practice Quiz | Coursera
New Tab
coursera.org/learn/sql-data-science/quiz/gWDiC/practice-quiz/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks
Back

Practice Quiz

Practice Quiz • 15 min • 5 total points

4. Which of the following is the correct order for accessing relational databases using Python?

1 / 1 point

○ create statements, connect.

○ create and execute SQL statements, connect, close connection.

○ create, execute Python statements, connect, close connection.

● connect, create and execute SQL statements, close connection.

✓ **Correct**

Correct.

5. Line magics: start with a single % (percent) sign and apply to a particular line in a cell.

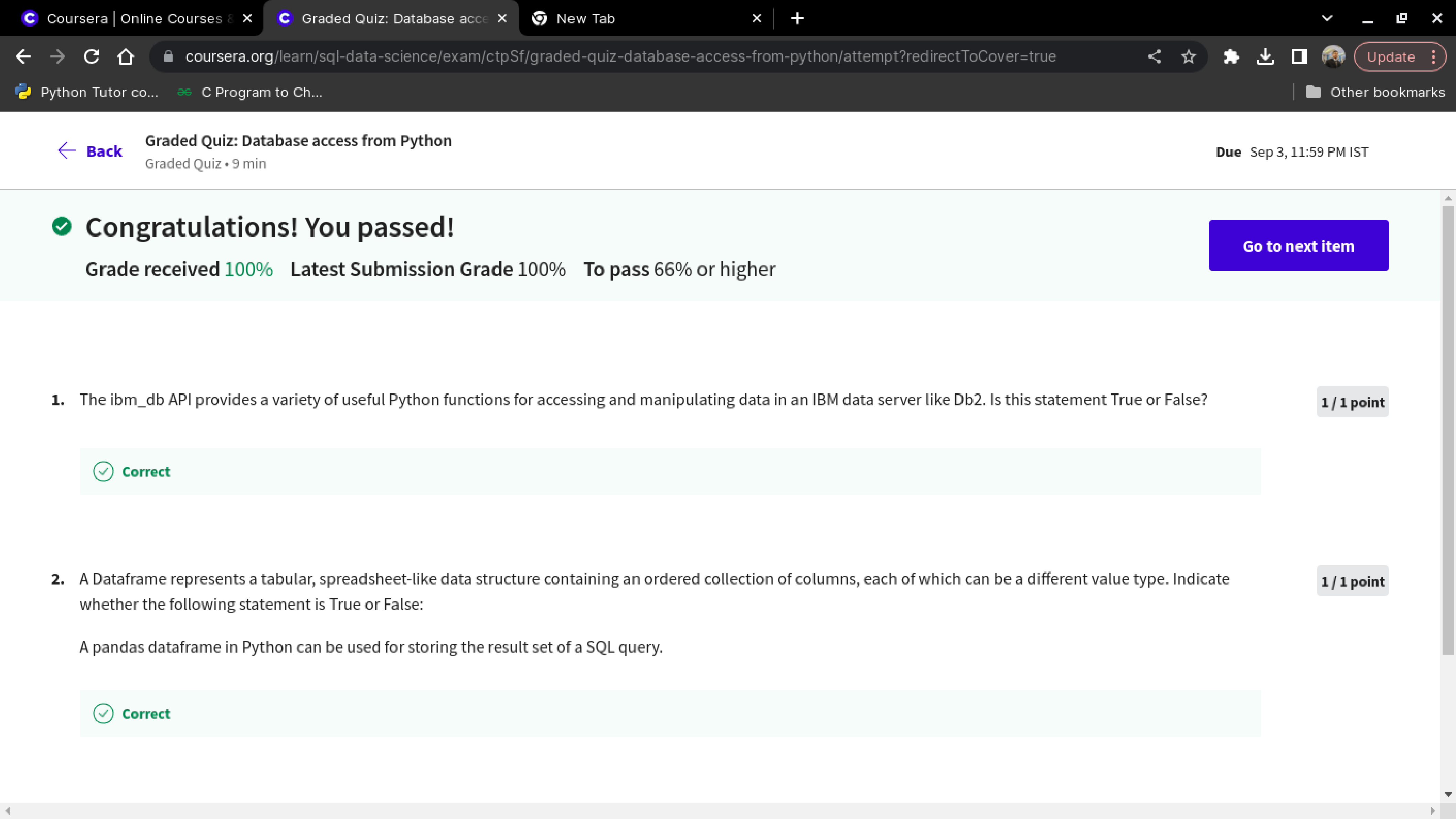
1 / 1 point

● True.

○ False

✓ **Correct**

Correct.



Coursera | Online Courses &
Graded Quiz: Database acce
New Tab
coursera.org/learn/sql-data-science/exam/ctpSf/graded-quiz-database-access-from-python/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks
Back
Go to next item

Graded Quiz: Database access from Python

Graded Quiz • 9 min

Due Sep 3, 11:59 PM IST

Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 66% or higher

1. The ibm_db API provides a variety of useful Python functions for accessing and manipulating data in an IBM data server like Db2. Is this statement True or False?

1 / 1 point

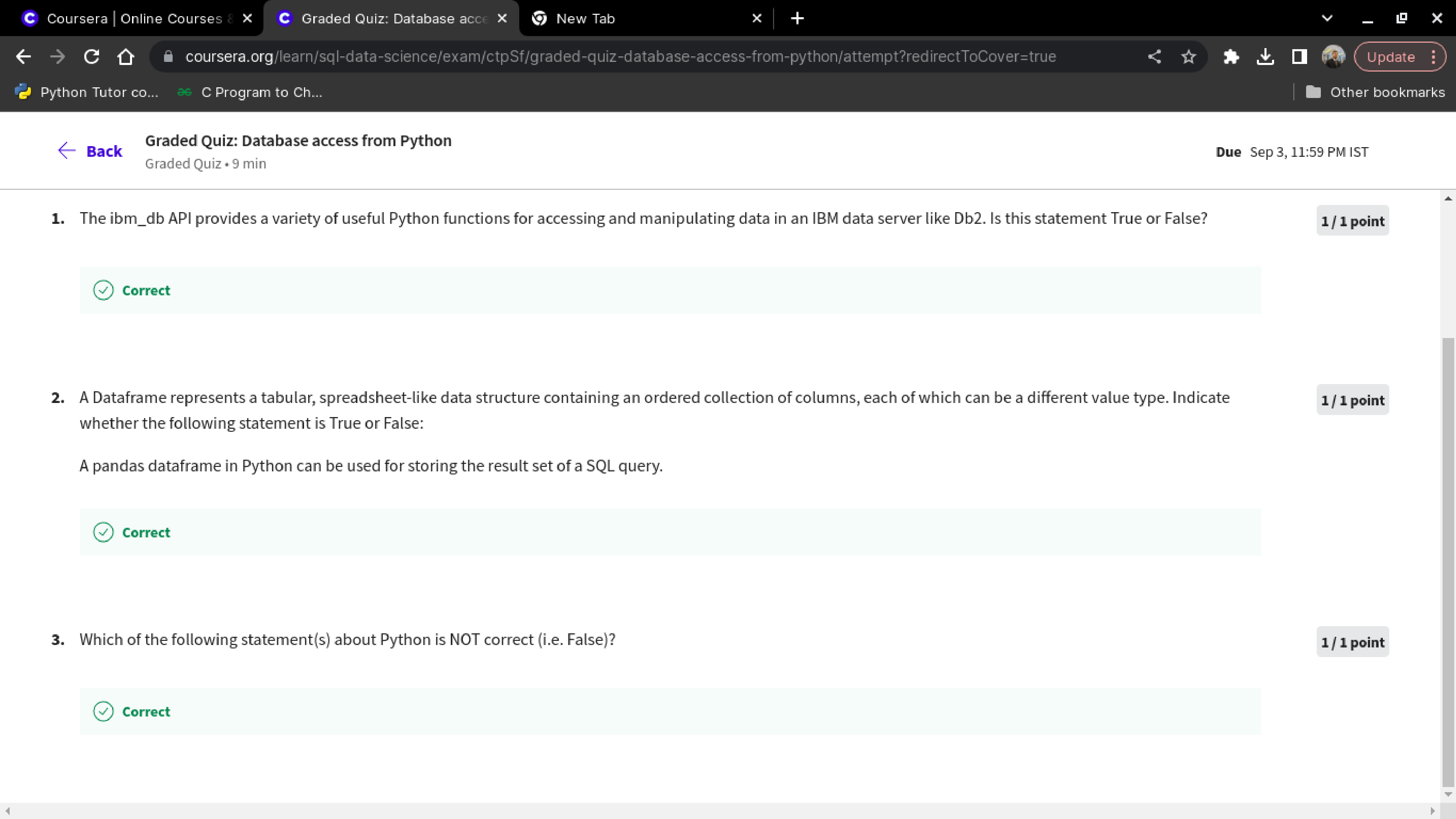
✓ **Correct**

2. A Dataframe represents a tabular, spreadsheet-like data structure containing an ordered collection of columns, each of which can be a different value type. Indicate whether the following statement is True or False:

A pandas dataframe in Python can be used for storing the result set of a SQL query.

1 / 1 point

✓ **Correct**


Coursera | Online Courses &
Graded Quiz: Database acce
New Tab
coursera.org/learn/sql-data-science/exam/ctpSf/graded-quiz-database-access-from-python/attempt?redirectToCover=true
Update
Python Tutor co...
C Program to Ch...
Other bookmarks
Back

Graded Quiz: Database access from Python

Graded Quiz • 9 min

Due Sep 3, 11:59 PM IST

1. The ibm_db API provides a variety of useful Python functions for accessing and manipulating data in an IBM data server like Db2. Is this statement True or False?

1 / 1 point

Correct

2. A Dataframe represents a tabular, spreadsheet-like data structure containing an ordered collection of columns, each of which can be a different value type. Indicate whether the following statement is True or False:

A pandas dataframe in Python can be used for storing the result set of a SQL query.

1 / 1 point

Correct

3. Which of the following statement(s) about Python is NOT correct (i.e. False)?

1 / 1 point

Correct